REMARKS

Claims 11-54 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejections:

The Office Action rejected claims 11, 13-19, 21, 23-28, 30-38, 40-46 and 48-54 under 35 U.S.C. § 103(a) as being unpatentable over Wang (U.S. Patent 6,292,936) in view of "The IR to VMx86 Translation Module Specification" by Chris Lattner, Dec. 1999 (hereinafter "Lattner").

Claims 12, 29, 39 and 47 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Lattner, and further in view of "The Principles of Computer Hardware, Third Edition" by Alan Clements, 2000 (hereinafter "Clements").

The Office Action rejected claims 20 and 22 under 35 U.S.C. § 103(a) as being unpatentable over Wang in view of Lattner and further in view of "Load-time Structural Reflection in Java" by Shigeru Chiba, June 2000.

Applicant respectfully traverses these rejections in light of the following remarks.

Regarding claim 11, the Examiner has stated that Wang teaches "generating a platform-independent representation of the one or more script language instructions." Applicant respectfully disagrees with the Examiner's interpretation of Wang.

Wang teaches a system wherein multiple intermediate sources are created from an original source document to enable multiple processors to interpret and process their respective intermediate source. Wang fails to teach generating a platform-independent representation of the one or more script language instructions.

In contrast, Wang teaches that when creating the intermediate source of a specific processor blocks from the original source that are not appropriate for the specific processor are replaced with synchronization and/or notification tokens to allow synchronization between the processors (Wang, column 3, lines 31-49). Wang gives as an example an input source containing HTML and VisualBasic Script. In this example, the HTML processor creates two intermediate sources, one for a Java Virtual Machine, intermediate source 200, and one for a VisualBasic Script interpreter, intermediate source 202. Intermediate source 202 for the VisualBasic Script interpreter contains the original VisualBasic Script instructions that it needs to process, and synchronization tokens where the original non-VisualBasic Script instructions were in the original source (Wang, column 3, lines 44-49, and Fig. 2). Wang does this so that each processor has an input file containing only those instructions it needs to process plus the synchronization information necessary for ordering the overall processing of the original source.

Likewise, intermediate source 200, for the JVM contains the HTML instructions for the JVM to process and notify and wait method invocation where the VisualBasic Script instructions were in the original source (Wang, column 3, lines 31 –43, and Fig. 2). Wang states this clearly with,"[t]he HTML Parser translates the remaining VisualBasic Script blocks in the original input source into notify method and wait method invocations" (Wang, column 4, line 65-column 5, line 1).

Therefore, Applicant asserts when Wang creates intermediate sources, he is not translating embedded script instructions, but merely *replacing* them with uniform synchronization calls or tokens. For example, every VisualBasic Script block is translated into notification and wait method invocations for the VJM, and every HTML block is translated into synchronization tokens for the VisualBasic Script interpreter. Clearly these replacements are not representations of the original script instructions. Therefore, Applicant asserts that Wang fails to teach generating a platform-independent representation of the one or more script language instructions.

Further regarding the teaching of Wang, the Examiner states, "[t]he interpreter inherently converts the script into an equivalent intermediate form." The Examiner's cited passage (Wang, column 1, lines 19-21) describes how an interpreter-based embedded scripting environment would generally convert an input source containing embedded script into an equivalent script format. However, the passage goes on to describe how an HTML text with included VisualBasic Script is converted into an equivalent VisualBasic Script "so that a VisualBasic Script interpreter only has to interpret the VisualBasic Script, and not the HTML text" (Wang, column 1, lines 26-28). Wang is clearly describing a system wherein the embedded script itself remains in the original script language (e.g. VisualBasic Script). The HTML text around the embedded script instructions has been translated and included in the VisualBasic Script and the VisualBasic Script instructions themselves remains unchanged. Hence, the interpreter described at column 1, lines 19-21 is not converting the script into an equivalent intermediate form as the Examiner has suggested.

Moreover, in the passage cited by the Examiner, Wang is describing the deficiency of a system that his invention teaches away from by splitting the original input source into multiple intermediate sources and not creating a single converted script. Therefore, Applicant submits that the Examiner is improperly attempting to rely on both the single interpreter based system and the multiple interpreter system that Wang introduces specifically to teach away from the single interpreter based system. It is improper to combine these two disparate teachings in Wang.

Regarding Applicants previous arguments that Lattner does not suggest representing script instructions as executable platform-independent programming object (see Applicants response mailed November 3, 2003), the Examiner states, "[i]t is not necessary to specify that Lattner teach representing script instructions, since Lattner already discloses the advantages of representing instructions as platform-independent programming objects." Applicant disagrees with the Examiner. As described in the Applicant's response to the previous office action, Lattner teaches an Instruction class which is designed to exist as a node in a linked list and generate a legal 80x86 assembly

language string for an intermediate language instruction associated with that particular node in the list. (page 2). The 80x86 assembly language instructions are of a wholly distinct and different character than embedded script instructions. Just because Lattner teaches the use of a Java-based Instruction class to represent 80x86 assembly instructions, does not suggest use of a Java-based Instruction class to represent script instructions. There is no suggestion in the prior art to apply a Lattner's Java-based Instruction class to represent anything other than 80x86 assembly instructions. Assembly instructions and script instructions are handled differently in computer systems. Prior art teachings in regard to assembly instructions do not automatically extend to script instructions.

Accordingly neither Wang nor Lattner, separately or in combination, suggest creating a platform-independent representation of embedded script instructions.

Furthermore, Applicant disagrees with the Examiner's assertion that it would have been obvious to combine Wang with Lattner. The Examiner argues that the motivation to combine Wang with Lattner is that Wang teaches an intermediate platform-independent representation and Lattner teaches a platform-independent representation of individual commands as programming language objects, that there is a benefit to storing commands as individual objects, and it would be beneficial to store the commands taught in Wang in objects taught be Lattner.

However, as shown above, Wang fails to teach an intermediate platform independent representation and Lattner does not suggest representing script instructions as executable platform-independent programming objects. Further, Wang's goal is to "allow multiple runtime processors to process their corresponding portion of the original input source" (Wang, column 1, lines 31 – 34). Applicant asserts that it would be counter-intuitive and counter-productive to translate the original script commands of Wang into executable platform-independent programming objects; thereby circumventing the need for multiple run-time processors and obviating the need for synchronization tokens, which are at the center of Wang's invention. Applicant respectfully reminds the

Examiner that "[i]f the modification would render the prior art invention being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification" (M.P.E.P § 2143.01 paragraph 10).

In light of the above remarks, Applicant asserts that the rejection of claim 11 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Likewise, independent claims 28, 38 and 46 recite subject matter similar to Applicant's claim 11, and are thus believed to patentably distinguish over the cited art for at least the reason given above.

Regarding claim 12, the Examiner contends that Clements teaches using the stack structure to hold instructions that are executed by popping the instructions off of the stack. Applicant disagrees with the Examiner's interpretation of Clements. In fact, Clements clearly fails to teach using a stack structure to hold *instructions*.

Clements teaches the storing of data on a stack and further teaches that the 68K family of processors includes instructions that manipulate and use the *contents* of the stack. Clements teaches, "a computer can *transfer data between memory and the stack*, and perform monadic operations on the top item of the stack, or dyadic operations on the top two items of the stack" (Emphasis added) (Clements, Section 6.5, paragraph 4). Clements further teaches the use of the stack to store the parameters and the return addresses of subroutines (Clements, Sections 6.5.2 and 6.5.4).

Applicant can find no reference in Clements regarding using a stack structure to hold instructions that are executed by popping the instructions off of the stack as the Examiner contends. Furthermore, Applicant notes that the Examiner has neglected to cite any particular passages in Clements supporting his contention.

The applicant also disagrees with the Examiner's assertion as to the obviousness of combining Wang, as modified by Lattner, with Clements.

As shown above, Wang and Lattner fail to teach the method of Claim 11. Further, Clements fails to teach using a stack structure to hold instructions. Lattner teaches the use of an Instruction class to encapsulate intermediate instructions in a linked list (page 2) while Clement teaches a hardware stack (page 3). A linked list is a wholly different type of structure and incompatible with a hardware stack. Modifying the representation in Lattner to be stack-based instead of a linked list would be directly counter to the teachings of Lattner, and thus an improper modification.

Therefore, in light of the above remarks, Applicants assert that the rejection of claim 12 is not supported by the cited art and withdrawal of the rejection is respectfully requested. Similar remarks as discussed above in regard to claim 12 apply to claims 29, 39, and 47.

Regarding claim 21, the Examiner takes official notice that removing methods and fields from a program object is well known, especially when methods and fields become outdated, and hence no longer have any function. However, even if removing methods from a program object may be well known *in other circumstances*, removing methods of a program object is not well known as part of generating a platform-independent programming language representation of a sequence of script language instructions. It is not enough for the Examiner to state that something is well known. As the Federal Circuit stated in *In re Kotzab*, 55 USPQ2d 1313, 1316 (Fed. Cir. 2000):

> Most if not all inventions arise from a combination of old elements. However, identification in the prior art of each individual part claimed is insufficient to defeat patentability of the whole claimed invention.

The Examiner has not cited any reference that suggests removing methods of a program object as part of generating a platform-independent programming language representation of a sequence of script language instructions. Thus, Applicant respectfully requests the removal of the rejection of claim 21.

Regarding claim 23, the Examiner takes official notice that removing methods and fields from a program object is well known, especially when methods and fields

become outdated, and hence no longer have any function. However, even if removing fields from a program object may be well known *in certain other circumstances*, removing fields of a program object is not well known as part of generating a platform-independent programming language representation of a sequence of script language instructions as the Examiner has asserted. The Examiner has not cited any reference that suggests removing fields of a program object as part of generating a platform-independent programming language representation of a sequence of script language instructions. Thus, Applicant respectfully requests the removal of the rejection of claim 23.

Further regarding claims 21 and 23, Applicant respectfully reminds the Examiner that merely stating that individual aspects of a claimed invention are well known does not render the combination well known without some objective reason to combine the individual teachings. *Ex parte Levengood*, 28 USPQ2d 1300. The Examiner did not provide any such reasoning. Therefore, the rejection is further improper. As the Court of Appeals for the Federal Circuit recently explained in *In re Sang Su Lee*, Docket No. 00-1158 (Fed. Cir. January 18, 2002), conclusory statements such as those provided by the Examiner that a claim limitation is well known or common knowledge do not fulfill the Examiner's obligation. "Deficiencies of the cited references cannot be remedied by the [Examiner's] general conclusions about what is 'basic knowledge' or 'common sense.'" *In re Zurko*, 59 USPQ2d 1693, 1697 (Fed. Cir. 2001).

Applicant also asserts that numerous ones of the other dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and notice to that effect is requested.

If any extension of time (under 37 C.F.R. § 1.136) is necessary to prevent the above referenced application from becoming abandoned, Applicant hereby petitions for such extension. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-60300/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Fee Authorization Form authorizing a deposit account debit in the amount of $ for fees ( ).

☐ Other:

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: April 8, 2004